УДК 681.3

АЛГЕБРА АЛГОРИТМИКИ И ТРАНСФОРМАЦИОННАЯ СВОДИМОСТЬ СХЕМ АЛГОРИТМОВ И ПРОГРАММ

А.С. Мохница

Институт программных систем НАН Украины,
03187, Киев, проспект Академика Глушкова, 40.
E-mail: mohnitsa@isofts.kiev.ua

Кратко охарактеризована алгебра алгоритмики и ее прототипов. Демонстрируется трансформационная сводимость структурной схемы алгоритма к неструктурной посредством инструментария трансформации схем алгоритмов и программ.

The brief characteristics of algebra algorithmics and its prototypes is given. Transformational reduceness of structural schema of algorithm to nonstructural using the toolkit for transformation of schemas of algorithms and programs is demonstrated.

Введение

В настоящее время на Западе актуальна проблематика порождающего программирования (Generative Programming, GP) (синтез программ). В Украине первая алгебра программ была построена В.М. Глушковым в рамках работ по системам алгоритмических алгебр (САА) [1] в середине 60-х годов, за несколько лет до того, как концепция подобной алгебры была предложена Э. Дейкстрой. Более того, еще в 1959 году в [2] Л.А. Калужнин предложил алгебру граф-схем как инструмент для математизации программирования, которая и стала отправной точкой для создания САА. В 1982 году была создана система синтеза программ МУЛЬТИПРОЦЕССИСТ [3, 4], в качестве входного языка была использована параллельная модификация САА – САА-М.

Современное состояние исследований украинской алгебро-кибернетической школы в области алгебры алгоритмики, восходящей к работам Глушкова и Калужнина, отражено в [5].

Среди направлений, развиваемых на Западе в русле концепции алгебры программ на данный момент, отметим алгебраическую алгоритмику (АА) [6, 7] и ментальное программирование (Intentional Programming, IP) – одна из ветвей GP [8]. В какой-то мере их можно назвать прототипами алгебры алгоритмики [9].

Цель работы показать, что отказ от использования языков программирования [8], предлагаемый авторами порождающего программирования для обеспечения правильности синтезируемых программ (в том числе и неструктурных), является ошибочным.

Работа имеет такую структуру. В разделе 1 дается краткая характеристика алгебры алгоритмики (<АА>) и ее прототипов. Раздел 2 посвящен инструментарию трансформации схем алгоритмов и программ. Трансформационная сводимость структурной схемы алгоритма к неструктурной посредством упомянутого инструментария демонстрируется в разделе 3.

1. Алгебра алгоритмики и ее прототипы

1.1. <АА> положена в основу алгебраической теории алгоритмов, в рамках которой осуществляется разработка средств представления, накопления, конструирования и классификации алгоритмических знаний, относящихся к различным предметным областям, в частности, к задачам символьной мультиобработки (сортировка, поиск, языковое процессирование [5]).

С <АА> связаны три формы представления (спецификации или алгеброалгоритмические модели) алгоритмов [5]:

аналитическая (формульная) – представление алгоритма в виде формулы в выбранной алгебре алгоритмов, удобна для трансформации, в частности для улучшения по выбранным критериям (память, быстродействие и др.);

текстовая (естественно-лингвистическая) – представление алгоритма на естественном (понятном пользователю) языке в терминах выбранной предметной области, удобна для диалогового проектирования правильных алгоритмов и программ на разных входных языках;

визуальная (графовая, граф-схемная) – представление алгоритма в виде граф-схем Калужнина [5], характеризуется в первую очередь наглядностью в процессе диалогового проектирования и, кроме того, является промежуточным звеном при переходе к UML-диаграммам.

Именно алгеброалгоритмические модели положены в основу инструментария <АА> [10–12]. Непосредственно с использованием моделей связаны методы (метаправила) проектирования алгоритмов и

программ [5]: восходящий (свертка, абстрагирование), нисходящий (развертка, детализация), комбинированный (переинтерпретация).

Одним из основных является метаправило трансформации, позволяющее выполнять преобразование аналитического представления алгоритмов и программ на основе применения аппарата тождеств, квазитождеств и соотношений, характеризующих свойства операций используемой алгебры алгоритмов, с целью улучшения по выбранным критериям.

Кроме того, метаправило переинтерпретации обеспечивает возможность переноса полученных алгоритмов на другие предметные области.

Отметим простоту, высокоуровневость и взаимодополняемость формализмов предлагаемых <АА> по сравнению с традиционными языками программирования (ЯП), что вместе с теорией клонов [5] и использованием метаправил обеспечивает надежный фундамент для проектирования и синтеза программ в объектно-ориентированных и распределенных (GRID) [13] средах.

1.2. АА представляет собой формализованный подход к описанию алгоритмов доказательства теорем в области классических алгебр (кольца, группы, факториалы и т.п.). Для реализации основных алгебраических структур и концепций в АА используется язык программирования АДА. Для других ЯП данный подход предлагает создавать специализированные конвертеры, что представляет собой нетривиальную задачу. В <АА> данная проблема отсутствует, так как инструментарий предлагает несколько целевых языков, причем при необходимости их список может быть пополнен.

Существенный недостаток АА состоит в отсутствии схемного (формульного) представления для алгоритмов, что не дает возможности для применения аппарата алгебраических преобразований для приведения к каноническому виду, качественного улучшения по выбранным критериям.

1.3. Концепция IP связана с формализацией программирования и базируется на моделировании семейств программных систем таким образом, что «по конкретным техническим требованиям можно автоматически получить специализированный и оптимизированный промежуточный или конечный продукт из элементарных, многократно используемых компонентов реализации с помощью базы знаний о конфигурациях» [8]. Кроме того, IP предназначено для облегчения и ускорения конструирования кода ЯП различного уровня и прикладного назначения. Эта цель определяет инструментарий IP, предназначенный для построения прикладных абстракций и развития средств их поддержки в связи с ориентацией на многообразие областей приложений.

Нельзя не отметить схожесть парадигм и инструментальных средств IP и <АА>: оперирование с абстракциями, расширяемость входных языков, применение метапрограммирования и графического представления программного кода, повторное использование компонентов, ориентированность на семейство систем в предметной области и т.д.

Несмотря на внешнюю схожесть IP и <АА>, основным отличием является прежде всего несхемный характер абстракций первого, что как и в случае с АА, исключает возможность применения алгебраических методов для качественного улучшения алгоритмов по тем или иным критериям. Кроме того, абстракции IP не тождественны абстракциям <АА>, представляющим три взаимосвязанные и взаимодополняющие формы спецификации алгоритма и дающим более полную информацию о нем.

2. Инструментарий трансформации схем алгоритмов и программ

В рамках работы над инструментальными средствами синтеза алгоритмов и программ в объектных средах [5] с помощью аппарата <АА> был спроектирован и реализован инструментарий трансформации схем алгоритмов и программ (Трансформатор) [5, 11].

Основным назначением Трансформатора является преобразование аналитических спецификаций алгоритмов (формул в различных алгебрах) с использованием соотношений и тождеств соответствующей алгебры с целью улучшения по выбранным критериям, кроме того, Трансформатор может быть использован для доказательства гипотез в рамках <АА> (например, эквивалентности булевых функций в алгебре логики [11]). Внешний вид Трансформатора показан на рис. 1.

В основу функционирования Трансформатора положен алгоритм последовательной статической декомпозиции ДЕК/С [5]. Во время статической декомпозиции форма (левая или правая часть тождества) накладывается на цепочку (преобразуемую формулу), или ее уже зафиксированную подцепочку с разбиением на составляющие, служащие значениями параметров формы. Полученные в результате декомпозиции значения переменных формы присваиваются соответствующим переменным противоположной части (правой или левой части) тождества, которая заменяет найденную подцепочку.

Процесс трансформации начинается с загрузки выражения, подлежащего преобразованию, и базы тождеств и соотношений, которые будут использованы в процессе работы.

Далее пользователю предлагается меню тождеств для применения к трансформируемому алгоритму с последующим выбором правой или левой его части и/или редактированием управляющего контекста. Если выбранное тождество применимо к преобразуемому выражению, то результат (проинтерпретированная левая или правая часть равенства) отображается в соответствующих полях, иначе выводится сообщение о неприменимости данного тождества при существующем контексте.

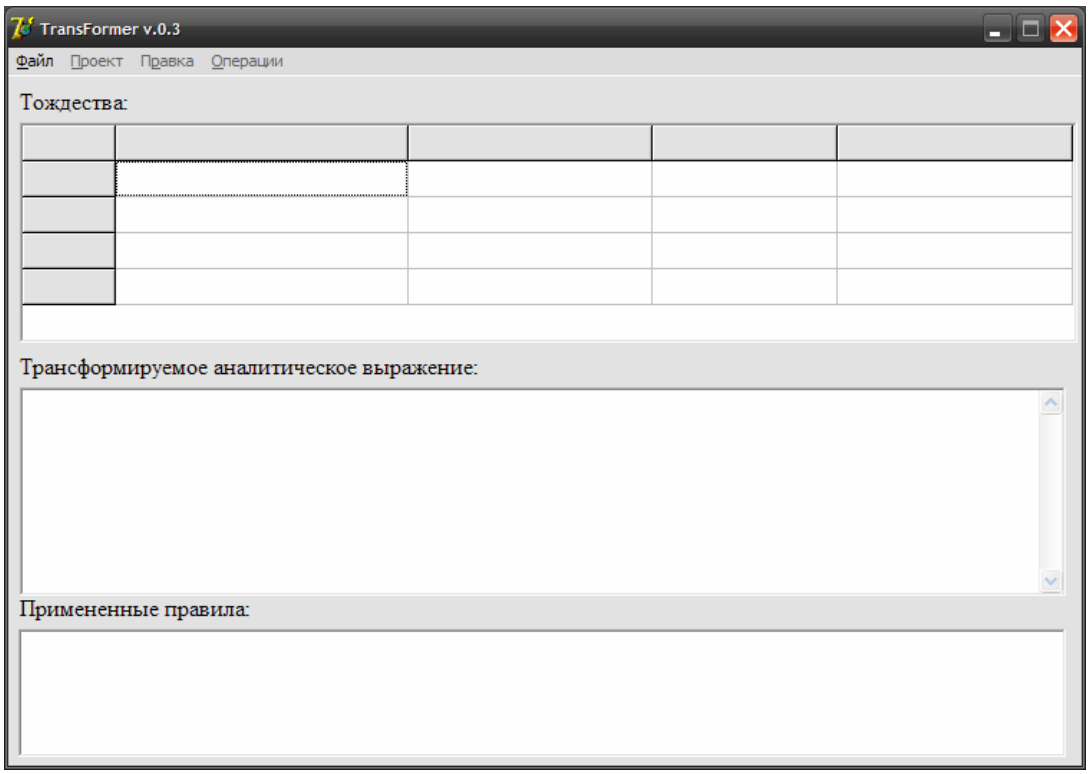

Рис. 1. Внешний вид инструментальных средств трансформации схем алгоритмов и программ

3. Трансформационная сводимость схем алгоритмов и программ

Продемонстрируем пошагово трансформационную сводимость структурной схемы алгоритма в АД к неструктурной посредством упомянутого инструментария трансформации схем алгоритмов и программ.

Рассмотрим алгебру неструктурных схем алгоритмов – алгебру Янова (АЯ) [5].

Система АЯ = <{АНС, $L(2)$}; СИГН'>,

где АНС – совокупность неструктурных схем,

L(2) – совокупность различных булевых функций,

СИГН' – сигнатура, состоящая из композиции $A*B$ и операции неструктурного перехода $\Pi(u, F)$, а также дизъюнкции, конъюнкции и отрицания.

Суть $\Pi(u, F) = \Pi(u) \downarrow m$, F є АНС, состоит в передаче управления на метку m при истинном условии u, иначе выполняется следующий за ним оператор. Таким образом, оператор $\Pi(u) \downarrow m$ аналогичен оператору Go to в языках программирования.

Производной от введенной операции, при $u = 1$, служит операция безусловного перехода $\Pi \downarrow m$. Также к числу производных операций АЯ относятся альтернатива и цикл, входящие в сигнатуру алгебры Дейкстры (АД), а также цикл типа DO-WHILE:

$$([u]\ A, B) = \Pi(u) \downarrow m\ B * \Pi \downarrow m'\ m: A * m': E,$$

$$\{[u]\ A\} = m: \Pi(u) \downarrow m'\ A * \Pi \downarrow m\ m': E,$$

$$\{A\ [u]\} = m: A * \Pi(НЕ(u)) \downarrow m\ E, \quad (1)$$

$$\{A\ [u]\} = \{[u]\ A\}, \quad (2)$$

где $A = A'$, либо A отличается от A' наличием меток.

Таким образом, в АЯ представима произвольная структурная схема, порожденная в АД, тем самым справедливо включение АСС є АНС, где АСС – множество схем, представимых в АД, АНС – множество неструктурных схем, представимых в АЯ.

Проиллюстрируем с помощью Трансформатора процесс преобразования структурной схемы в АД в неструктурную посредством применения введенных равенств. В качестве примера опишем в АД алгоритм ПРИНТ функционирования абстрактного принтера [5]:

ПРИНТ ::=

ПЕЧАТЬ_ПЕРВОЙ_СТРОКИ *

* {[КОНЕЦ_ФАЙЛА] ПЕЧАТЬ_ТЕКУЩЕЙ_СТРОКИ};

ПЕЧАТЬ_ПЕРВОЙ_СТРОКИ ::= Р($У_1$) * ПЕЧ*

* {[КОНЕЦ_СТРОКИ] Р($У_1$) * ПЕЧ} * ПЕРЕНОС;

ПЕЧАТЬ_ТЕКУЩЕЙ_СТРОКИ ::=

ПЕЧ * {[КОНЕЦ_СТРОКИ] Р($У_1$) * ПЕЧ}* ПЕРЕНОС;

Предполагается, что в начальном состоянии указатель $У_1$ расположен непосредственно слева от первого символа файла, выводимого на печать, а при переходе на очередную строку он устанавливается непосредственно на ее первый символ.

Перейдем с помощью таблицы интерпретаций к неинтерпретированной схеме *S*(ПРИНТ), отражающей структуру приведенной схемы ПРИНТ:

$$S(\text{ПРИНТ}) ::= A * B*\{[u]\ A * B\} * C * \{[u']\ B * \{[u]\ A * B\}*C\}.$$

Таблица

A	Р($У_1$)
B	ПЕЧ
u	КОНЕЦ_СТРОКИ
C	ПЕРЕНОС
u	КОНЕЦ_ФАЙЛА

Преобразуем схему S(ПРИНТ) в ее неструктурный эквивалент, применяя введенные ранее равенства.

Шаг 1. Применив равенство (2) в направлении справа налево к S(ПРИНТ), получим (рис. 2):

$$A * \{B * \{[u]\ A * B\} * C\ [u']\}.$$

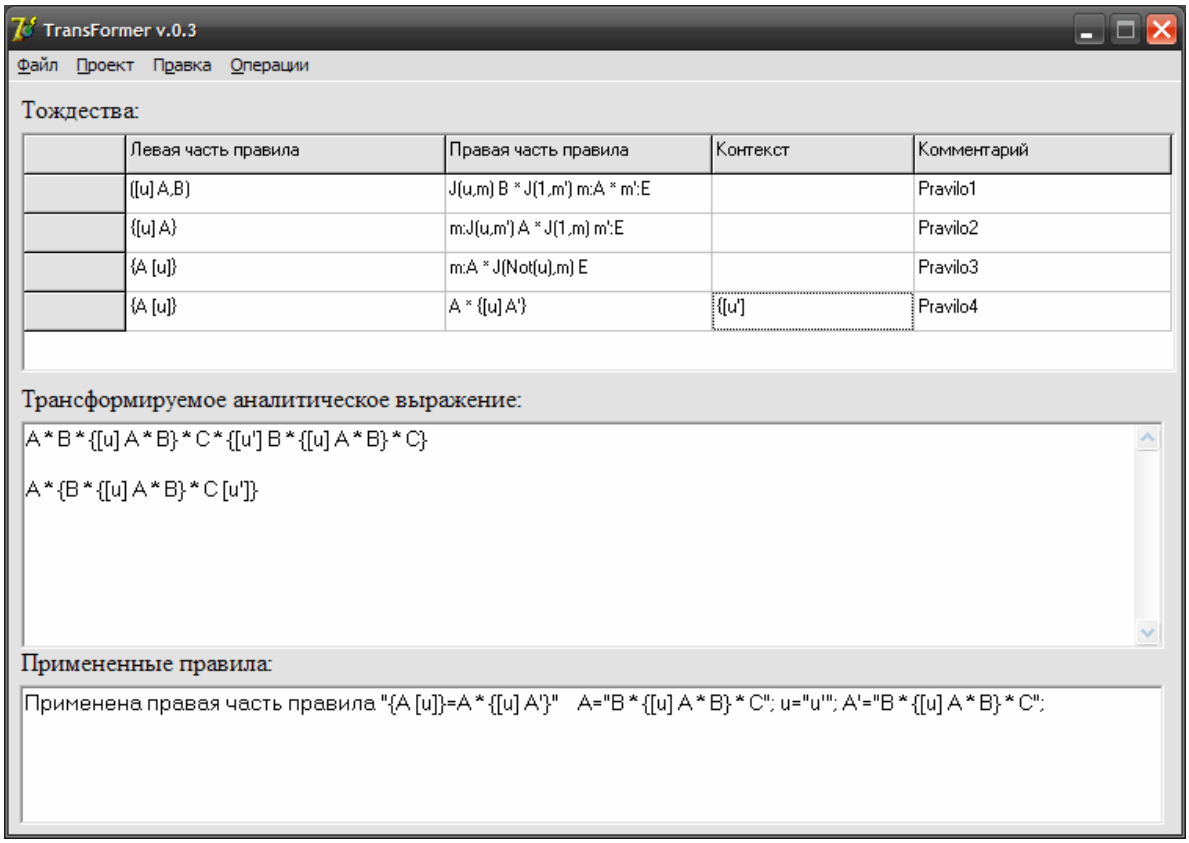

Рис. 2. Первый шаг трансформации

Шаг 2. Применим равенство (1) в направлении слева направо (рис. 3):

$$A * m': B * \{[u]\ A * B\} * C * \Pi(\text{НЕ}(u')) \downarrow m'\ E.$$

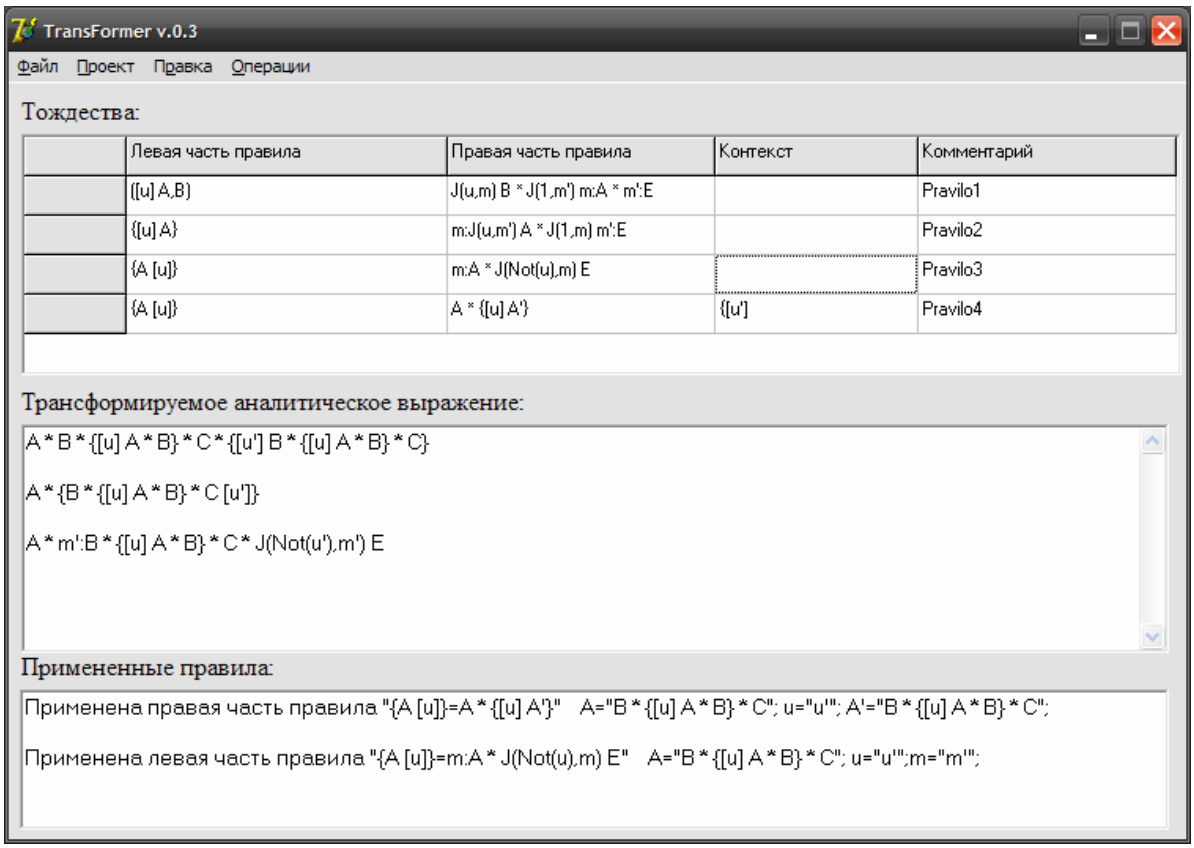

Рис. 3. Второй шаг трансформации

Шаг 3. Применим равенство (2) в направлении справа налево (рис. 4):

$$\{A * m' : B\,[u]\} * C * \Pi(\text{НЕ}(u')) \downarrow m'\ E.$$

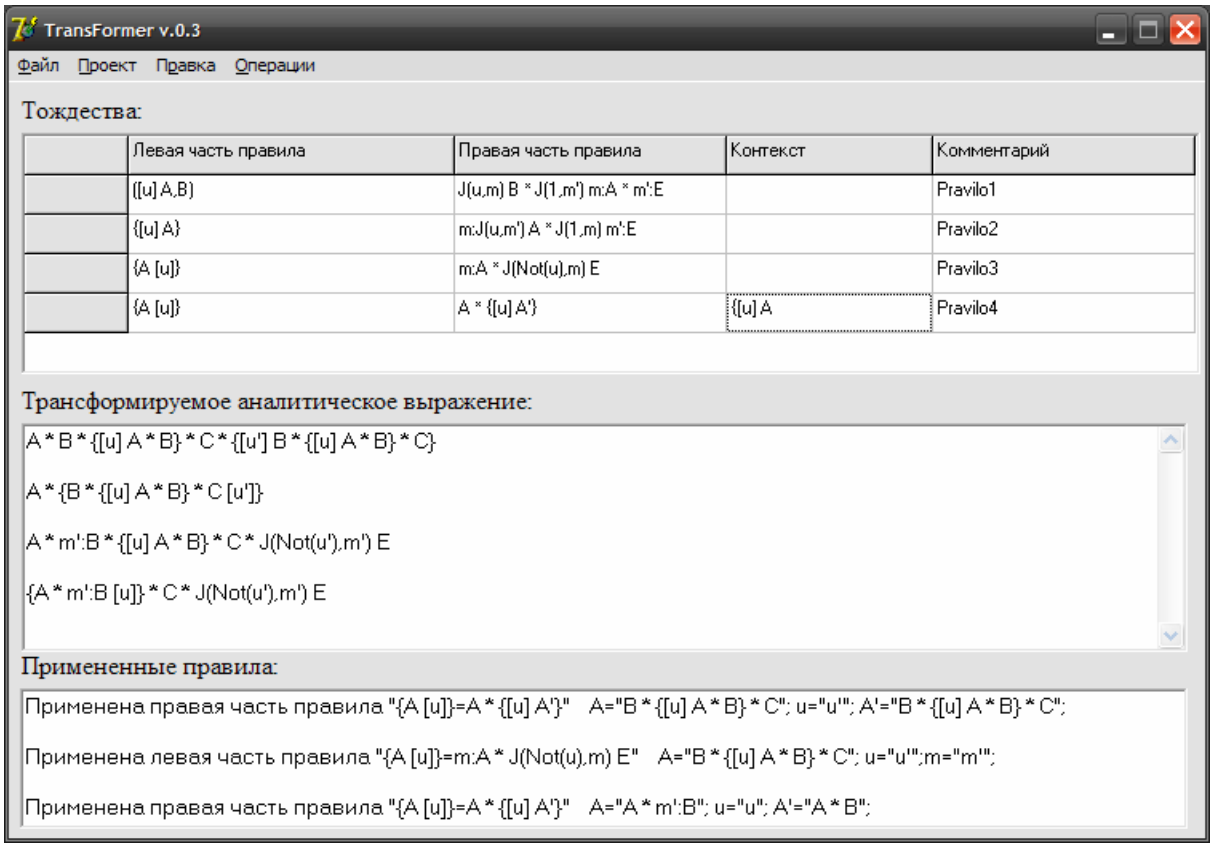

Рис. 4. Третий шаг трансформации

Шаг 4. Используем равенство (1) в направлении слева направо (рис. 5):

$$m\text{: } A * m'\text{: } B * \Pi(\text{НЕ}(u)) \downarrow m\ C * \Pi(\text{НЕ}(u')) \downarrow m'\ E.$$

Проинтерпретировав с помощью таблицы данную неструктурную схему, получим эквивалентное представление алгоритма ПРИНТ в АЯ:

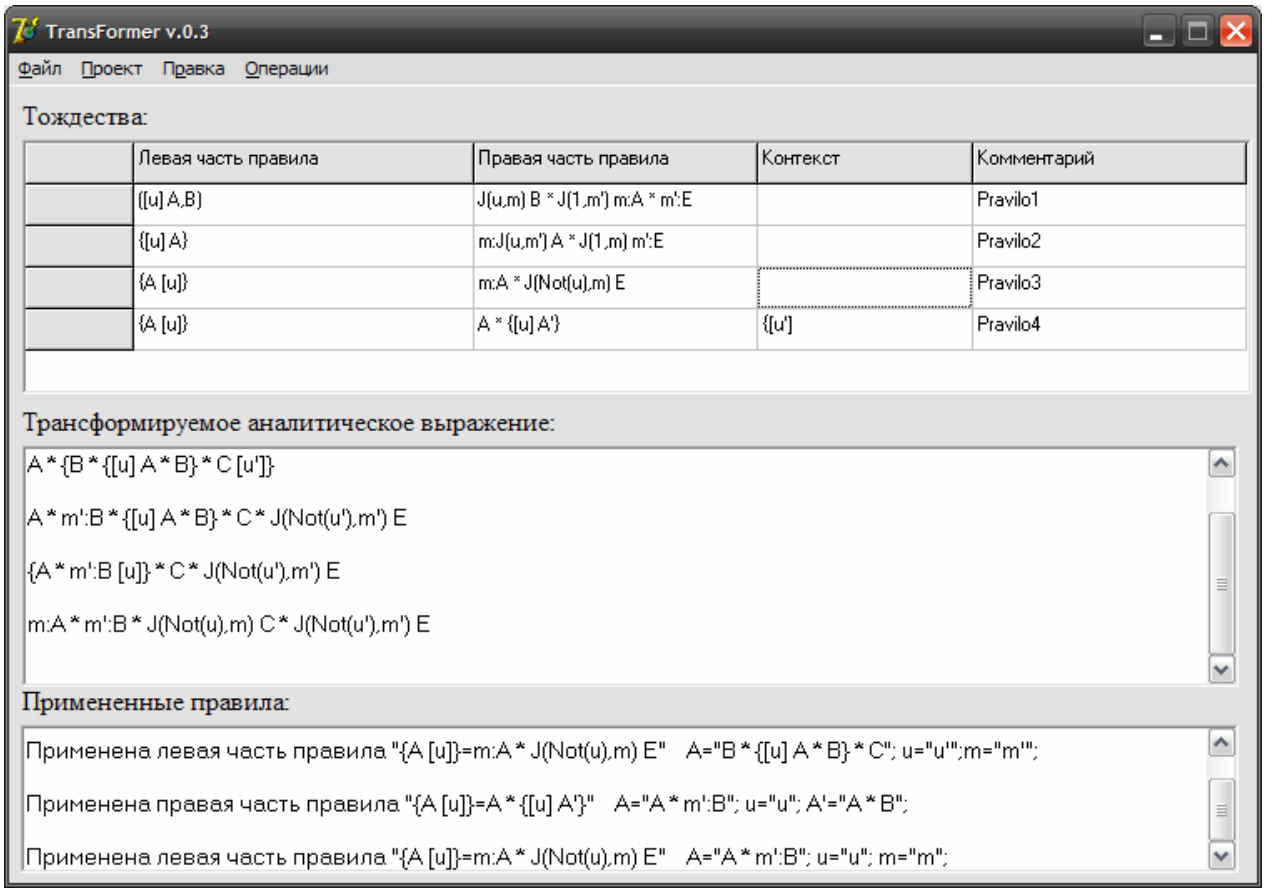

Рис. 5. Четвертый шаг трансформации

$$\text{ПРИНТ\_АЯ} ::= m\text{: } P(\text{У}_1) * m'\text{: ПЕЧ} * \Pi(\text{НЕ}(\text{КОНЕЦ\_СТРОКИ})) \downarrow m\ \text{ПЕРЕНОС} * \Pi(\text{НЕ}(\text{КОНЕЦ\_СТРОКИ})) \downarrow m'\ E.$$

Отметим, что полученная схема более компактна, а кроме того и улучшена по памяти. Подчеркнем, что данная схема является существенно неструктурной (не может быть представлена в АД, а только в алгебре Глушкова [5]) – с зацеплением двух циклов.

Таким образом справедливо следующее утверждение.

Теорема. Схема ПРИНТ трансформационно сводима к ПРИНТ_АЯ.

Справедливость данного утверждения следует из того, что равенства (1) и (2) являются тождественными преобразованиями.

Теорема доказана.

Эквивалентность выполненных с помощью Трансформатора преобразований обеспечивает правильность полученной схемы. Помимо этого, инструментарий <АА> гарантирует правильность алгоритма еще на этапе конструирования. Поэтому призыв разработчиков IP к отказу от ЯП (в том числе и неструктурных) из-за того, что не возможно обеспечить правильность синтезируемой программы является ошибочным.

Заключение

Кратко охарактеризованы направления алгебраической алгоритмики, ментального программирования и алгебры алгоритмики. Показаны их общие черты и отличия. Приведено описание инструментальных средств трансформации как составной части инструментария алгебры алгоритмики. Продемонстрирован процесс пошагового преобразования структурной схемы в неструктурную с сохранением функциональной эквивалентности посредством инструментария трансформации схем алгоритмов и программ.

1. *Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л.* Алгебра. Языки. Программирование. – Киев: Наук. думка, 1-е изд., 1974. – 327 с.; 2-е изд., перераб., 1978. – 318 с.; 3-е изд., перераб. и доп, 1989. – 376 с.
2. *Калужнин Л.А.* Об алгоритмизации математических задач // Проблемы кибернетики. – 1959. – Вып. 2. – С. 51–69.
3. *Ющенко Е.Л., Цейтлин Г.Е., Грицай В.П., Терзян Т.К.* Многоуровневое структурное проектирование программ: Теоретические основы, инструментарий. – М.: Финансы и статистика, 1989. – 208 с.
4. *Ющенко Е.Л., Цейтлин Г.Е., Галушка А.В.* Алгебро-грамматические спецификации и синтез структурированных схем программ // Кибернетика. – 1989. – № 6. – С. 5–16.
5. *Андон Ф.И., Дорошенко А.Е., Цейтлин Г.Е., Яценко Е.А.* Алгеброалгоритмические модели и методы параллельного программирования. – Киев: Академпериодика, 2007. – 634 с.
6. *Ноден П., Китте К.* Алгебраическая алгоритмика (с упражнениями и решениями). – М.: Мир, 1999. – 720 с.
7. *Цейтлин Г.Е., Мохница А.С.* Что такое алгебраическая алгоритмика? // // Проблемы программирования. Спец. выпуск по материалам 4-й Междунар. научн.-практ. конф. по программированию УкрПРОГ'2004. – Киев: ИПС НАН Украины, 2004. – № 2-3. – С. 52–57.
8. *Чарнецки К., Айзнекер У.* Порождающее программирование: методы, средства и приложения. – Питер: 2005. – 736 с.
9. *Дорошенко А.Е., Захария Л.М., Цейтлин Г.Е.,* Алгебраическое проектирование программ: алгоритмы, объекты, инструменты // Проблемы программирования. – 2007. – № 2. – С. 5–14.
10. *Цейтлин Г.Е., Яценко Е.А.* Элементы алгебраической алгоритмики и объектно-ориентированный синтез параллельных программ // Математические машины и системы. – 2003. – № 2. – С. 64–76.
11. *Яценко Е.А., Мохница А.С.* Инструментальные средства конструирования синтаксически правильных параллельных алгоритмов и программ // Проблемы программирования. Спец. выпуск по материалам 4-й Междунар. научн.-практ. конф. по программированию УкрПРОГ'2004. – Киев: ИПС НАН Украины, 2004. – № 2–3. – С. 444–450.
12. *Мохница А.С.* Инструментальные средства трансформации схем алгоритмов и программ // Проблемы программирования. Спец. выпуск по материалам 5-й Междунар. научн.-практич. конф. по программированию УкрПРОГ'2006. – Киев: ИПС НАН Украины, 2006. – № 2-3. – С. 377–382.
13. *Дорошенко А.Е., Алистратов О.В., Тырчак Ю.М., Розенблат А.П., Рухлис К.А.* Системы Grid-вычислений – перспектива для научных исследований // Проблемы программирования. – 2005. – № 1. – С. 14–38.